

ipd4400malltpmTES-10

**Defense Information Infrastructure (DII)
Common Operating Environment (COE)**

**Programming Manual (PM)
for the
Latitude-Longitude-Time (LLT)
Observation API (MALLT) Segment
of the
Tactical Environmental Support System Next Century
[TESS(NC)]
Meteorology and Oceanography (METOC) Database**

Document Version 4.4

5 February 1999

**Prepared for:
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA**

**Prepared by:
Integrated Performance Decisions
Middletown, RI**

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE

ipd4400malltpmTES-10

Table of Contents

1	SCOPE.....	1
1.1	Identification	1
1.2	System Overview.....	1
2	REFERENCED DOCUMENTS	5
2.1	Government Documents	5
2.2	Non-Government Documents.....	7
3	MALLT OVERVIEW	9
4	SEGMENT DEVELOPMENT	15
4.1	Writing Applications Using the MALLT APIs	16
4.1.1	Ingesting Observations Into the Database	16
4.1.2	Retrieving a Catalog of LLT Observations in the Database	18
4.1.3	Retrieving LLT Observations From the Database	21
4.1.4	Updating an LLT Observation Record in the Database	25
4.1.5	Deleting LLT Observations From the Database	26
4.1.6	Wildcards and NULL Values.....	28
4.1.7	Checking for NULL Values	30
4.2	Building Applications Using the MALLT Libraries	30
4.2.1	Makefile Using the Dynamic MALLT Libraries on HP-UX	30
4.2.2	Makefile Using the Static MALLT Libraries on HP-UX	31
4.2.3	Makefile Using the Dynamic Link MALLT Libraries on Windows NT	32
4.2.4	Makefile Using the Static MALLT Libraries on Windows NT	33
5	CUSTOMIZING SEGMENTS	35
6	NOTES	37
6.1	Glossary of Acronyms.....	37
	Appendix A - Valid Values, WMO Symbolic Code, and Units for MALLT Structures.....	A-1

List of Tables

3-1	METOC Database Observation Data Types	10
-----	---	----

List of Figures

1-1	TESS(NC) METOC Database Conceptual Organization	3
3-2	MDLLT Observation Classes	13

1 SCOPE

1.1 Identification

This Programming Manual (PM) describes the use of the Latitude-Longitude-Time (LLT) Observation Application Program Interface (API) (MALLT) segment, Version 4.3.0.0, of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database. The MALLT segment provides APIs for the storage, retrieval, and manipulation of point METOC observations. These are observations that are taken at a point in space and time, and are therefore characterized by latitude, longitude, and time. This software is designed to run under the Defense Information Infrastructure (DII) Common Operating Environment (COE) release 3.1 on a Hewlett-Packard computer running HP-UX 10.20 or a personal computer running the Microsoft Windows NT 4.0 operating system with Service Pack 3.

1.2 System Overview

The software described in this document forms a portion of the METOC Database component of the TESS(NC) Program (Navy Integrated Tactical Environmental Subsystem (NITES) Version I). On 29 October 1996, the Oceanographer of the Navy issued a TESS Program Policy statement in letter 3140 Serial 961/6U570953, modifying the Program by calling for five seamless software versions that are DII COE compliant, preferably to level 5.

The five versions are:

- NITES Version I The local data fusion center and principal METOC analysis and forecast system (TESS(NC))
- NITES Version II The subsystem on the Joint Maritime Command Information System (JMCIS) or Global Command and Control System (GCCS) (NITES/Joint METOC Segment (JMS))
- NITES Version III The unclassified aviation forecast, briefing, and display subsystem tailored to Naval METOC shore activities (currently satisfied by the Meteorological Integrated Data Display System (MIDDS))
- NITES Version IV The Portable subsystem composed of independent Personal Computers (PCs)/workstations and modules for forecaster, satellite, communications, and Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance (IC4ISR) functions (currently the Interim Mobile Oceanographic Support System (IMOSS))
- NITES Version V Foreign Military Sales (currently satisfied by the Allied Environmental Support System (AESS))

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps weather forecasters and tactical planners. NITES I provides these users with METOC data, products, and applications necessary to support the warfighter in tactical operations and decision making. NITES I provides METOC data and products to NITES I and II applications, as well as non-TESS(NC) systems requiring METOC data, in a heterogeneous, networked computing environment.

The TESS(NC) Concept of Operations and system architecture require that the METOC Database be distributed both in terms of application access to METOC data and products and in terms of physical location of the data repositories. The organizational structure of the database is influenced by these requirements, and the components of this distributed database are described below.

In accordance with DII COE database concepts, the METOC Database is composed of six DII COE-compliant *shared database* segments. Associated with each shared database segment is an API segment. The segments are arranged by data type as follows:

<u>Data Type</u>	<u>Data Segment</u>	<u>API Segment</u>
Grid Fields	MDGRID	MAGRID
LLT Observations	MDLLT	MALLT
Textual Observations and Bulletins	MDTXT	MATXT
Remotely Sensed Data	MDREM	MAREM
Imagery	MDIMG	MAIMG
Climatology Data	MDCLIM	MACLIM

A typical client-server installation is depicted in Figure 1-1. This shows the shared database segments residing on a DII COE SHADE database server, with a NITES I or II client machine hosting the API segments. Communication between API segments and shared database segments is accomplished over the network using American National Standards Institute (ANSI)-standard Structured Query Language (SQL).

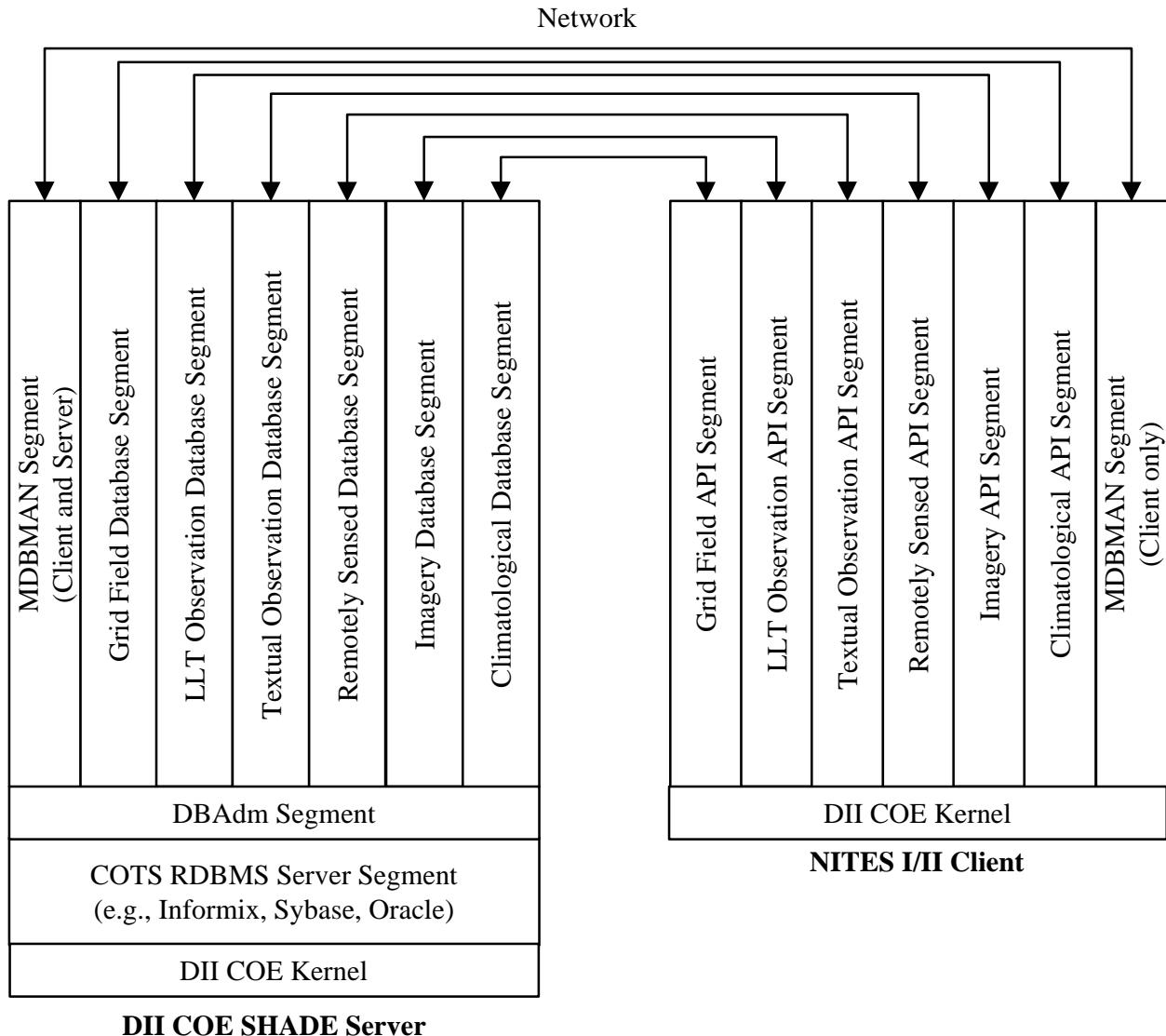


Figure 1-1. TESS(NC) METOC Database Conceptual Organization

The MALLT segment deals with point observations. These include surface weather observations (hourlies, specials, synoptic observations, METAR reports, Terminal Aerodrome Forecasts (TAFs), etc.), upper air observations (e.g., radiosonde reports, aircraft observations), and ocean soundings (bathythermograph, sound velocity profiles, etc.). For upper air and ocean soundings, the database may also store data derived from the original soundings in the form of upper air profiles and ocean profiles.

(This page intentionally left blank.)

2 REFERENCED DOCUMENTS

2.1 Government Documents

STANDARDS

MIL-STD-498
5 December 1994

Software Development and Documentation

SPECIFICATIONS

DII COE I&RTS
July 1997

Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification, Preliminary, Version 3.0

Unnumbered
30 September 1997

Software Requirements Specification for the Tactical Environmental Support System/Next Century [TESS(3)/NC] Meteorological and Oceanographic (METOC) Database, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

Unnumbered
5 December 1997

Performance Specification (PS) for the Tactical Environmental Support System/Next Century TESS(3)/NC (AN/UMK-3)

OTHER DOCUMENTS

DII.COE.DocReqs-5
29 April 1997

Defense Information Infrastructure (DII) Common Operating Environment (COE) Developer Documentation Requirements, Version 1.0

Unnumbered
30 September 1997

Database Design Description for the Tactical Environmental Support System/Next Century [TESS(3)/NC] Meteorological and Oceanographic (METOC) Database, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

ipd4400malltrmTES-10
5 February 1999

Application Program Interface Reference Manual (APIRM) for the Latitude-Longitude-Time (LLT) Observation API (MALLT) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database

Office of the Assistant Secretary of Defense (Command, Control, Communications, and Intelligence), Washington, DC

DoD 8320.1-M-1
19 November 1996

Data Standardization Procedures (Draft)

Commander, Naval Meteorology and Oceanography Command

COMNAVMETOCOMINST 3141.2 *Surface METAR Observations Users Manual*

COMNAVMETOCOMINST 3144.1D *Ship Weather Observations Manual*

Department of the Air Force, Headquarters Air Weather Service, Scott AFB, ILL

AWSR 105-2
24 August 1990

Weather Communications Policies and Procedures

Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

Unnumbered
20 June 1997

Tactical Environmental Data System (TEDS) Release 3.5 Observation/Profile Data Applications Program Interface (API) User's Guide

Office of the Federal Coordinator for Meteorological Services and Supporting Research, Washington, DC

FMH-2
December 1988

Federal Meteorological Handbook No. 2: Surface Synoptic Codes

FMH-10
December 1988

Federal Meteorological Handbook No. 10: Meteorological Rocket Observations

FMH-11
June 1991

Federal Meteorological Handbook No. 11: Doppler Radar Meteorological Observations

FMH-1
December 1995

Federal Meteorological Handbook No. 1: Surface Weather Observations and Reports

2.2 Non-Government Documents

World Meteorological Organization, Geneva, Switzerland

WMO-386
1992

Manual on the Global Telecommunications System

WMO-306
1995

Manual On Codes

(This page intentionally left blank.)

3 MALLT OVERVIEW

The MALLT segment provides APIs used to access point (LLT) data in the TESS(NC) METOC Database. The schema for storing these data is defined by the shared LLT Observation Database segment (MDLLT). Both of these segments require the Informix Relational Database Management System (RDBMS), Version 7.22 (for HP-UX machines) or 7.23 (for Windows NT machines). Both segments run in the DII COE release 3.1, hosted on the following machines and operating systems:

- Tactical Advanced Computer, TAC-3 (HP 750/755)/TAC-4 (HP J201), Operating System: HP-UX 10.20
- IBM-Compatible PC, Operating System: Microsoft Windows NT 4.0, with Service Pack 3

The MALLT uses the following environment variables related to the Informix installation:

- `INFORMIXSERVER` Identifies the Informix server, typically set to `online_coe`
- `INFORMIXDIR` Path to the Informix software, typically `/opt/informix` on HP systems and `C:\informix` on Windows NT systems

The path specified in the `INFORMIXDIR` variable should also be included in the system's PATH variable.

The MALLT segment is delivered as both archive and runtime libraries on both platforms, with filenames as follows:

<u>Library Type</u>	<u>HP-UX Filename</u>	<u>Windows NT Filename</u>
Archive	<code>libMALLTAPI.a</code> <code>libMALLTDsmgr.a</code> <code>libMALLTKernel.a</code> <code>libMALLTUtils.a</code>	<code>malltapi.lib</code> <code>malltdsmgr.lib</code> <code>malltkernel.lib</code> <code>malltutils.lib</code>
Runtime	<code>libMALLTAPI.sl</code> <code>libMALLTDsmgr.sl</code> <code>libMALLTKernel.sl</code> <code>libMALLTUtils.sl</code>	<code>malltapi.dll</code> <code>malltdsmgr.dll</code> <code>malltkernel.dll</code> <code>malltutils.dll</code>

The runtime libraries are typically installed in the directory /h/MALLT/bin on HP systems and C:\h\MALLT\bin on Windows NT systems. To use the runtime libraries in Windows NT, the directory containing the .dll files must be in the system's PATH.

The archive libraries are typically installed in the directory /h/MALLT/lib on HP systems and C:\h\MALLT\lib on Windows NT systems.

The individual API methods are described in the APIRM referenced in Section 2. Section 4 of this document provides instructions and programming examples for the use of the APIs.

The LLT Observation database was designed using an object/relational hybrid method. The observation data are modeled as classes derived from a common root class. However, the methods are not associated with each class; they are associated with the LLT Observation database in general. The purpose of the LLT Observation APIs is to manage all aspects of decoded meteorological observations from various sources (AWN, GTS, SMOOS) and formats (BUFR, WMO Textual Formats). Table 3-1 describes the decoded messages that MDLLT is designed to handle.

Table 3-1. METOC Database Observation Data Types

Type	Description	WMO Alphanumeric Code	WMO BUFR Sequence
Fixed Surface Station Synoptic Report	Synoptic report from surface stations reported at regular intervals.	FM-12	3-07-001 3-07-002 3-07-003 3-07-004 3-07-005 3-07-006 3-07-007 3-07-008 3-07-009
Ship Synoptic Report	Synoptic report from ships reported at regular intervals.	FM-13	3-08-004
Mobile Surface Station Synoptic Report	Synoptic report from mobile land stations reported at regular intervals.	FM-14	No equivalent
METAR/SPECI	Aviation Routine Weather Report and Aviation Selected Special Weather Report.	FM-15, FM-16	3-07-011

Table 3-1. METOC Database Observation Data Types

Type	Description	WMO Alphanumeric Code	WMO BUFR Sequence
Fixed Buoy Report	Report from fixed ocean buoys.	FM-18	3-08-001 3-08-002
Drifting Buoy Report	Report from drifting ocean buoys.	FM-18	3-08-003
Upper Air Winds Report at pressure levels from fixed land station	Upper air report from a surface station that reports only wind information at standard and significant isobaric levels that was taken from a fixed land station.	FM-32	3-09-003 3-09-004
Upper Air Winds Report at pressure levels from sea station	Upper air report from a surface station that reports only wind information at standard and significant isobaric levels that was taken from a sea station.	FM-33	3-09-012
Upper Air Winds Report at pressure levels from a mobile land station	Upper air report from a surface station that reports only wind information at standard and significant isobaric levels that was taken from a mobile land station.	FM-34	3-09-016 3-09-019
Upper Air Temperature Report at pressure levels from a fixed land station	Upper air report from a surface fixed land station that reports wind and temperature information at standard and significant isobaric levels.	FM-35	3 09 005 3 09 006 3 09 007 3 09 008
Upper Air Temperature Report at pressure levels from a ship station	Upper air report from a ship station that reports wind and temperature information at standard and significant isobaric levels.	FM-36	3 09 013 3 09 014 3 09 017 3 09 018
Upper Air Temperature Report at pressure levels from a drop sonde	Upper air report from a sonde dropped from a balloon or aircraft station that reports wind and temperature information at standard and significant isobaric levels.	FM-37	No equivalent

Table 3-1. METOC Database Observation Data Types

Type	Description	WMO Alphanumeric Code	WMO BUFR Sequence
Upper Air Temperature Report at pressure levels from a fixed land station	Upper air report from a surface fixed land station that reports wind and temperature information at standard and significant isobaric levels.	FM-38	No equivalent
Upper Air Winds Report at heights from a surface station	Upper air reports from a surface station that reports only wind information at geo-potential heights.	FM-39	3-09-001 3-09-002 3-09-003 3-09-004
Upper Air Winds Report at heights from a ship	Upper air reports from a ship station that reports only wind information at geo-potential heights.	FM-40	3-09-011
Report from Aircraft	Report from aircraft.	FM-41, FM-42 ICAO Aireps	03-11-001
Aerodrome Forecast	Report and forecast from airfields.	FM-51	
Bathy Report	Report of a bathythermal observation.	FM-63	3-15-001
TESAC	Temperature, salinity, and current reports from a sea station.	FM-64	3-15-002

At the root class, minimal information is available within each class:

- Time
- Latitude
- Longitude
- Observation Type
- Observation Subtype
- Originator
- Classification
- Source Communication Channel
- Air Temperature
- Wind Speed
- Wind Direction
- Pressure.

Leaf node classes model decoded messages and contain all data that was received from the decoded message to allow “re-assembly” of messages. Figure 3-2 illustrates the class hierarchy for LLT observation data.

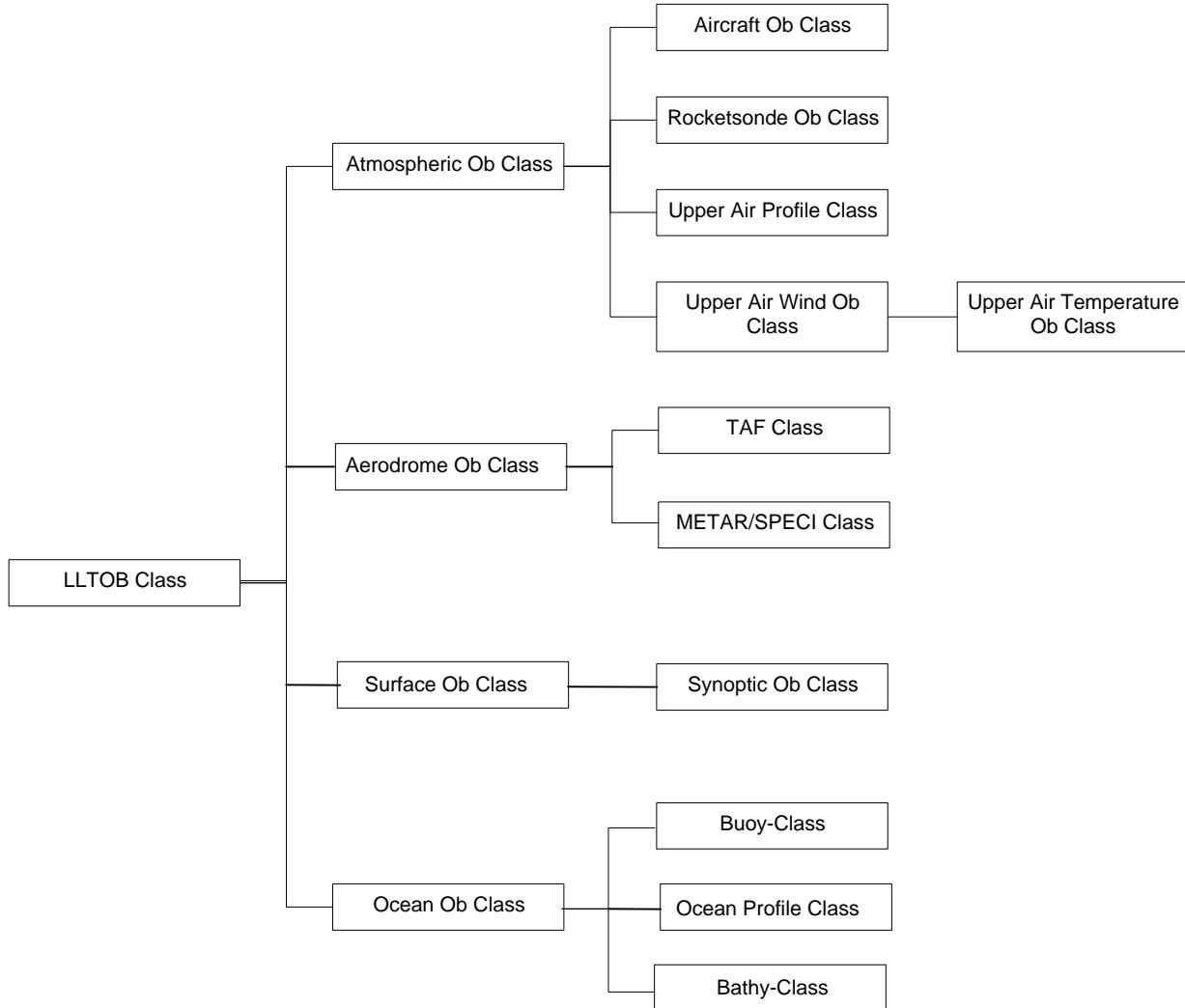


Figure 3-2. MDLLT Observation Classes

(This page intentionally left blank.)

4 SEGMENT DEVELOPMENT

Programming applications to access LLT observation and bulletin data are straightforward. The MALLT segment provides interfaces to:

- Connect to the TESS(NC) METOC Database (MALLTConnect and MALLTRemoteConnect)
- Set the active connection (MALLTSetConnection)
- Ingest an LLT observation into the database (MALLTIgest)
- Retrieve a catalog listing of LLT observations meeting specified criteria from the database (MALLTCatalog)
- Retrieve selected LLT observation data from the database (MALLTGetByQuery)
- Retrieve a single LLT observation from the database (MALLTGetByID)
- Retrieve IDs of stations that reside in a specified area from the database (MALLTGetStationByArea)
- Retrieve the geographic area based on a station ID from the database (MALLTGetStationByID)
- Update an LLT observation data record in the database (MALLTUpdateByID)
- Delete an LLT observation data record from the database (MALLTDeleteByID)
- Purge observations from the database based on selection criteria (MALLTDeleteByQuery)
- Free the linked lists returned by MALLTCatalog and MALLTGetByQuery (MALLTFreeLL)
- Disconnect from the database (MALLTDisconnect and MALLTRemoteDisconnect).

Each of these methods is described in detail in the MALLT API Reference Manual, referenced in Section 2.

4.1 Writing Applications Using the MALLT APIs

This section shows the use of the MALLT APIs to perform common data access tasks. In each case, an overview of the actions to be performed is provided, along with a code example and a discussion of pertinent programming concerns.

In all cases, note that the MALLTConnect or MALLTRemoteConnect method must be called to connect the application to the database before any other operations may be performed. MALLTDisconnect or MALLTRemoteDisconnect should be called to disconnect the application from the database at the end of the session. These methods only need to be called once for each database needed by an application. If an application handles connections to more than one database, switching between connections is done by using the MALLTSetConnection method.

All structures must be initialized through use of calloc, memset or the appropriate MALLTset<>2Null function. Failure to initialize a structure could cause unpredictable results. Garbage in a structure could cause a query to fail or ingestion of an observation to be rejected.

The MALLTRET structure is used to return status information from each MALLT method. See Section 3.4.1.2 of the APIRM for details of the information returned in this structure.

4.1.1 Ingesting Observations Into the Database

The MALLTIgest method is used to add an observation to the database. It takes as input a pointer to a MALLTOBSERVATION structure containing a decoded observation. Certain observation structure fields must have non-null values and must be within a given range of values. Appendix A of this document is a table that maps fields within structures to valid values. Out-of-range values will cause the API to return a constraint violation error, and the observation will not be ingested. Mappings of structure fields to WMO symbolic codes are also contained within this table.

```
/**************************************************************************/  
/* Purpose: This sample program demonstrates how to ingest a pirep into */  
/* the database. The basic steps are: */  
/* */  
/*      1. Connect to the database using the MALLTConnect API */  
/*      2. Insert the observation. */  
/*      3. Disconnect from the database by calling the */  
/*          MALLTDisconnect API */  
/* */  
/* Tips: Use the "malltcommon.h" and "malltupperairob.h" header files */  
/*       to map fields between structures and their associated */  
/*       database tables. */  
/*       The API sets the obRef field within the ingested ob to the table */  
/*       and object id of the ingested ob. */  
/* */
```

```
/*
 * Obtype and subtype values are defined in the "malltypes.h"      */
/* Responsible Organization:                                         */
/*          Integrated Performance Decisions, Monterey Sector (RLJones)   */
/*                                                               */
/*********************************************************/
#include <stdio.h>
#include "MALLTAPI.h"

main()
{
    MALLTOBSERVATION anOb;
    MALLTRET mr;
    /* Clear the ob structure */
    memset(&anOb,0,sizeof(MALLTOBSERVATION));

    /* Set up the type/subtype of the observation */
    anOb.obType.sObType = MALLT_AIRCRAFT_OB;      /* Defined in malltypes.h*/
    anOb.obType.sObSubType = MALLT_PIREP;        /* Defined in malltypes.h*/

    /* Set up the "general conditions" */
    anOb.gc.rsAirTemperature = -10;

    /* Indicates that wind speed and direction were not reported */
    anOb.gc.rsWindSpeed= MALLTsetFloat2Null();
    anOb.gc. sWindDirection = MALLTsetShort2Null();
    anOb.gc. rsPressure = 740.0;
    anOb.gc. rsAltitude = 10000;

    /* Set up the geolocation of the ob */
    anOb.geoLoc.rsLatitude = 31.00;
    anOb.geoLoc.rsLongitude = -100.0;

    /* Set ob time to current time */
    anOb.reportTime.lTime = time(NULL);

    /* Set up the PIREP specific fields */
    strcpy(anOb.ob.pirep.szRemarks,"Sure is bumpy up here!");
    anOb.ob.pirep.rsMoisture = 10;
    strcpy(anOb.ob.pirep.szPhaseOfFlight,"LVR");
    anOb.ob.pirep.szAirframeIcing[0]=' ';
    anOb.ob.pirep.szMoistureType[0]= '1';
    anOb.ob.pirep.szNavSystem[0]= 0;
    anOb.ob.pirep.sTurbulence=2;
    anOb.ob.pirep.sTurbulenceTop=1000;

    /* Set up aircraft identifier */
    strcpy(anOb.reporterID.szSiteName,"UAL300");

    /* Set up receipt method */
    strcpy(anOb.reportInfo.szReceiptMethod,"MEDS");

    /* Connect to the database */
    mr = MALLTConnect();
    if (!mr.nStatus)
    {
        if(!mr.nStatus) mr = MALLTIngest(&anOb);
    }
    if(mr.nStatus) /* Nonzero value indicate an error */
    {
```

```

        printf("MALLTRET: status %d ",mr.nStatus);
        printf("SQLSTATE %s \n",mr.szSQLState);
        printf("Error Message: %s \n",mr.szErrorMessage);
    }
    else
    {
        printf("Table:           %s\n",anOb.obRef.szTableName);
        printf("Object ID:       %d\n",anOb.obRef.nObjectID);
    }
    mr = MALLTDisconnect(); /* Disconnect from database */
}

```

4.1.2 Retrieving a Catalog of LLT Observations in the Database

The MALLTCatalog method queries the active database for observations meeting specific criteria. The criteria are contained in the input MALLTCATALOGQUERY structure. The context of each field is the same for all observations except for TAFs. The nMinTime and nMaxTime are used to describe a range of valid times, whereas for other observations, it is used to describe a range of actual observation times.

MALLTCatalog returns a linked list of **MALLTOBCATALOG** structures for all observations except for TAFs. TAFs are returned in **MALLTFCSTCATALOG** structure. The function return value is a **MALLTRET** status structure. The linked list returned by **MALLTCatalog** must be freed when the data it contains are no longer needed by calling the function **MALLTFreeLL** in order to avoid memory leaks.

The catalog query structure is common for all Observations with the exception of TAFs. TAFs have additional fields that may be used for query criteria.

```
*****
/*Purpose: This example performs a catalog for upper air temperature obs */
/*          from fixed land stations, within a given area for stations */
/*          that have reported within the last 2 hours. */
*/
/* Query Criteria is:
*/
/*      OBTYPE      = MALLT_UPPERAIR_TEMP
*/
/*      SUBTYPE     = MALLT_FIXED_LAND
*/
/*      SITE ID     = wildcard
*/
/*      NORTH LAT   = 50.0
*/
/*      SOUTH LAT   = 20.0
*/
/*      WEST LON    = -120.0
*/
/*      EAST LON    = -80.0
*/
/*      MAX TIME    = current time
*/
/*      MIN TIME    = current time - 3600 (seconds in an hour)
*/
*/
/*      1. Connect to the database using the MALLTConnect API
*/
/*
/*      2. Retrieve a catalog of observations that meet the
/*          query criteria using a call to the MALLTCatalog API
*/
*/
```

```
/*
 *          3. Display the catalog information.           */
/*
 *          4. Call the MALLTFreeLL API to free the catalog   */
 *              linked-list before disconnecting           */
/*
 *          5. Disconnect from the database by calling the    */
 *              MALLTDisconnect API                         */
/*
 * Tips: Use the "malltcommon.h" and "malltupperairrob.h" header files   */
 *       to map fields between structures and their associated      */
 *       database tables.                                         */
/*
 * Obtype and subtype values are defined in the "mallttypes.h"        */
 *       file.                                              */
/*
 * Responsible Organization:                                           */
 *       Integrated Performance Decisions, Monterey Sector (RLJones) */
/*
*****include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "MALLTAPI.h"

*****M A I N
*****{/}

{
    MALLTOBCATALOGQUERY catQuery;
    PMALLTOBCATALOG pObCatalog=NULL;
    PMALLTFCSTOBBCATALOG pFObCatalog=NULL;
    MALLTLINKEDLIST ll,catList,*pCatList=NULL;
    MALLTRET malltRet;
    int nNumObs=0;
    /* Specify desired catalog query criteria */
    /* define desired observation type and subtype */
    catQuery.sObType = MALLT_UPPERAIR_TEMP; /* obtype: UPPER AIR TEMP */
    catQuery.sObSubType = MALLT_FIXED_LAND; /* subtype: FIXED LAND REPORT */
    /* Wildcard WMO ID */
    catQuery.nSiteID = MALLTsetInt2Null();
    /* wildcard the call sign */
    catQuery.szSiteName[0] = 0;
    /* Query within given area*/
    catQuery.geoArea.rsNorthLat = 80.0;
    catQuery.geoArea.rsSouthLat = -80.0;
    catQuery.geoArea.rsWestLon = -170.0;
    catQuery.geoArea.rsEastLon = -170.0;
    /* Set time range to be current time to current time - an hour */
    catQuery.nMinTime = time(NULL)-3600;
    catQuery.nMaxTime = time(NULL);
    /* Ignore Time that obs were ingested */
    catQuery.nMinIngestTime = MALLTsetInt2NULL();
    catQuery.nMaxIngestTime = MALLTsetInt2NULL();
    /* initialize the catalog list structure */
    memset(&catList,0,sizeof(MALLTLINKEDLIST));
```

```
*****  
/* Connect to the default "mdllt_db" database */  
*****  
malltRet = MALLTConnect();  
if(malltRet.nStatus)  
{  
    printf("MALLTRET: status %d ",malltRet.nStatus);  
    printf("SQLSTATE %s \n",malltRet.szSQLState);  
    printf("Error Message: %s \n",malltRet.szErrorMessage);  
}  
else  
{  
    printf("\nMALLTConnect succeeded.\n");  
    /* Call API with query for summary of obs that match criteria */  
    malltRet = MALLTCatalog( catQuery, &nNumObs, &catList );  
}  
if(malltRet.nStatus)  
{  
    printf("MALLTRET: status %d ",malltRet.nStatus);  
    printf("SQLSTATE %s \n",malltRet.szSQLState);  
    printf("Error Message: %s \n",malltRet.szErrorMessage);  
}  
else  
{  
    /* View each catalog */  
    printf("\nMALLTCatalog returned nNumObs = %d\n",nNumObs);  
    for (pCatList = &catList;  
         !pCatList && pCatList->data;  
         pCatList = pCatList->next)  
    {  
        pObCatalog = (PMALLTOBCATALOG)pCatList->data;  
        printf("\n-----\n");  
        printf(" CATALOG ENTRY (observation summary):\n");  
        printf("-----\n");  
        printf("Object ID : %d\n",pObCatalog->obRef.nObjectID);  
        printf("Table : %s\n",pObCatalog->obRef.szTableName);  
        printf("Ob Type id : %hd\n",pObCatalog->obType.sObType);  
        printf("Ob Type : %s\n",pObCatalog->szObTypeName);  
        printf("Ob Sub Type id : %hd\n",pObCatalog->obType.sObSubType);  
        printf("Ob Sub Type : %s\n",pObCatalog->szObSubTypeName);  
        printf("SiteName : %s\n",pObCatalog->szSiteName);  
        printf("ReceiptMethod : %s\n",pObCatalog->szReceiptMethod);  
        printf("Name ID : %d \n",pObCatalog->nSiteID);  
        printf("Flag : %d \n",pObCatalog->nDataCategory);  
        printf("nQualityIndicator : %d \n",pObCatalog->nQualityIndicator);  
        printf("Latitude : %f \n",pObCatalog->geoLoc.rsLatitude);  
        printf("Longitude : %f \n",pObCatalog->geoLoc.rsLongitude);  
        printf("Obtime(n) : %d \n",pObCatalog->baseTime.lTime);  
        printf("Obtime(c) : %s \n",pObCatalog->baseTime.szTime);  
        if(pObCatalog->obType.sObSubType == MALLT_TAF_REPORT)  
        {  
            /* Print TAF specific data. */  
            pFObCatalog = (PMALLTFCSTOBBCATALOG) pObCatalog;  
            printf("Obtime(n) : %d \n",pFObCatalog->begininvalidtime.lTime);  
            printf("Obtime(c) : %s \n",pFObCatalog->begininvalidtime.szTime);  
            printf("Obtime(n) : %d \n",pFObCatalog->endinvalidtime.lTime);  
        }  
    }  
}
```

```
        printf("Obtime(c)      : %s \n",pFObCatalog->endvalidtime.szTime);
    }
}
printf("\nCalling MALLTFreeLL to free the catalog linked list.\n");
MALLTFreeLL(&catList);
/*****************/
/* Disconnect from the "mdllt_db" database */
/*****************/
MALLTDisconnect();
printf("\nDisconnecting from the database.\n");
exit(0);
}
```

4.1.3 Retrieving LLT Observations From the Database

Two methods are provided for retrieving LLT observations:

- MALLTGetByQuery
- MALLTGetByID

The MALLTGetByQuery method takes as input a MALLTCATALOGQUERY structure containing criteria for observations to be retrieved. It returns a linked list of MALLTOBSERVATION structures containing all observations that matched the input criteria. The function return is a MALLTRET status structure. The linked list returned by MALLTGetByQuery must be freed when no longer needed.

The second method for retrieving LLT observations is MALLTGetByID. This method takes as input a MALLTOBREFERENCE structure identifying the observation to be retrieved, and returns a single MALLTOBSERVATION structure containing the retrieved data and a MALLTRET structure. This method requires knowledge of the reference data for the record to be retrieved, which is typically found by calling MALLTCatalog to retrieve a list of LLT observations meeting specific criteria. Note that if MALLTCatalog is called, the linked list it returns must be freed afterward.

MALLTGetByQuery Code Example

```
*****
/* Purpose:                                     */
/*   This is a sample program on how to use the MALLTGetByQuery API. */
/*                                                 */
/* Query criteria is:                           */
/*   OBTYPE      =  MALLT_UPPERAIR_OB           */
/*   SUBTYPE     =  MALLT_FIXED_LAND           */
/*   SITE ID     =  wildcard (Latitude/Longitude instead) */
/*   NORTH LAT   =  Wild card                  */
/*   SOUTH LAT   =  Wild card                  */
/*   WEST LAT    =  Wild card                  */
/*   EAST LAT    =  Wild card                  */
/*   MIN TIME    =  current - 1200*60*60  (seconds) */

*****
```

```
/*
 */
/* Tips: Use the "malltcommon.h" and "malltupperairrob.h" header files */
/* to map fields between structures and their associated */
/* database tables. */
*/
/*
*/
/* Responsible Organization: */
/* Integrated Performance Decisions, Monterey Sectors (RLJones) */
*/
*****
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "MALLTAPI.h"

*****
M A I N
***** */

void main()
{
    /* initialize the catalog query and list structures */
    int nNumObs = 0;
    MALLTLINKEDLIST *pOBList, obList;
    PMALLTObservation pOB;
    MALLTOBCATALOGQUERY catQuery;
    MALLTOBCATALOG *pObCatalog;
    MALLTRET mr;

    mr = MALLTConnect();
    CheckError(mr);
    if(!mr.nStatus)
    {

        memset(&catQuery, 0, sizeof(MALLTOBCATALOGQUERY));
        /*
        */
        /* Specify desired catalog query criteria */
        /*
        */
        /* define desired observation type and subtype*/
        catQuery.sObType = MALLT_UPPERAIR_TEMP;
        catQuery.sObSubType = MALLT_FIXED_LAND;

        /* wildcard the geographic search area */
        catQuery.geoArea.rsNorthLat = MALLTsetFloat2Null();
        catQuery.geoArea.rsSouthLat = MALLTsetFloat2Null();
        catQuery.geoArea.rsWestLon = MALLTsetFloat2Null();
        catQuery.geoArea.rsEastLon = MALLTsetFloat2Null();

        /* wildcard the time range limits */
        catQuery.nMinTime = time(NULL) - 12*60*60; /* current-12 hours */
        catQuery.nMaxTime = time(NULL); /* current time */
        /* Ignore Time that obs were ingested */
        catQuery.nMinIngestTime = MALLTsetInt2NULL();
        catQuery.nMaxIngestTime = MALLTsetInt2NULL();

        /* wildcard the station ID */
        catQuery.nSiteID = MALLTsetInt2Null();
```

```
/* wildcard the call sign */
catQuery.szSiteName[0] = 0;
/* Wildcard other fields */
catQuery.nDataCatagory = MALLTsetInt2Null();
catQuery.nQualityIndicator = MALLTsetInt2Null();
mr = MALLTGetByQuery ( catQuery,&nNumObs,&obList);
CheckError(mr);
for (pOBList = &obList;
     pOBList && pOBList->data && !mr.nStatus;
     pOBList = pOBList->next )
{
    pOB = (PMALLTOBSERVATION) pOBList->data;
    PrintUAT(pOB); /* See code common.h */
}
MALLTDDisconnect();
}
exit(0);
}
```

MALLTGetByID Code Example

```
*****
/*
/* Purpose: This program demonstrates how to use the MALLTGetByID API.
/*
/* Query Criteria is:
/*
/*          OBTYPE      =  MALLT_UPPERAIR_TEMP
/*          SUBTYPE     =  MALLT_FIXED_LAND
/*          SITE ID     =  wildcard (Latitude/Longitude instead)
/*          NORTH LAT   =  90.0      (This example specifies the whole
/*          SOUTH LAT   =  -90.0     world as the area to search)
/*          WEST LAT    =  -180.0
/*          EAST LAT    =  180.0
/*          MIN TIME    =  current - 1200*60*60 (seconds)
/*
/*          Obtype and subtype values are defined in the "malltypes.h"
/*
/* Responsible Organization:
/*          Integrated Performance Decisions, Monterey Sectory (RLJones)
/*
*****
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "MALLTAPI.h"

*****
M A I N
*****
```

```
void main()
{
    MALLTOBCATALOGQUERY  catQuery;
    MALLTOBCATALOG        *pObCatalog;
    MALLTOBSERVATION      theOb;
    int                  nNumObs = 0;
    MALLTLINKEDLIST       catList,*pCatList;
```

```
MALLTRET          mr;

/* initialize the catalog query and list structures */
memset(&catQuery,0,sizeof(MALLTOBCATALOGQUERY));
/* initialize the catalog list structure */
memset(&catList,0,sizeof(MALLTLINKEDLIST));
/*****************/
/* Connect to the default "mdl1t_db" database */
/*****************/
mr = MALLTConnect();
CheckError(mr);
if(!mr.nStatus)
{
    /* **** */
    /* Specify desired catalog query criteria */
    /* **** */
    /* define desired observation type and subtype */
    catQuery.sObType = MALLT_UPPERAIR_TEMP;
    catQuery.sObSubType = MALLT_FIXED_LAND;
    /* Query entire world alternate method is to set each geographic
       point to NULL using MALLTsetFloat2Null() */
    catQuery.geoArea.rsNorthLat = 90.0;
    catQuery.geoArea.rsSouthLat = -90.0;
    catQuery.geoArea.rsWestLon = -180.0;
    catQuery.geoArea.rsEastLon = 180.0;
    /* wildcard the time range limits */
    catQuery.nMinTime = time(NULL) - 1200*60*60; /* current-offset */
    catQuery.nMaxTime = time(NULL);           /* current time */
    /* wildcard the station ID */
    catQuery.nSiteID = MALLTsetInt2Null();
    /* wildcard the call sign */
    catQuery.szSiteName[0] = 0;
    /* **** */
    /* Call API with query for summary of obs that match criteria */
    /* **** */
    catQuery.nDataCatagory = MALLTsetInt2Null();
    catQuery.nQualityIndicator = MALLTsetInt2Null();
    mr = MALLTCatalog( catQuery, &nNumObs, &catList );
    CheckError(mr);
}

if(!mr.nStatus)
{
    /* **** */
    /* View each catalog entry & retrieve & view each ob referenced */
    /* **** */
    for (pCatList = &catList;
         pCatList && pCatList->data && !mr.nStatus;
         pCatList = pCatList->next )
    {
        printf("\nMALLTCatalog returned nNumObs = %d\n",nNumObs);
        ViewCatalog(pCatList->data);
        pObCatalog = (PMALLTOBCATALOG)pCatList->data;
        mr = MALLTGetByID(pObCatalog->obRef,&theOb);
        if(!CheckError(mr)) PrintUAT(&theOb);
    }
}
MALLTFreeLL(&catList); /* Catalog Linked List */
MALLTDisconnect();      /* Disconnect from the "mdl1t_db" database */
exit(0);
} /* end main */
```

4.1.4 Updating an LLT Observation Record in the Database

The MALLTUpdateByID is used to update an existing LLT observation record in the database. MALLTUpdateByID takes as inputs a MALLTOREFERENCE structure identifying the record to be updated and a pointer to a MALLTObservation structure containing the data with which the record is to be updated. The MALLTObservation structure must be completely filled in with all relevant data for the updated observation (there is no provision for updating only certain fields of the observation record while retaining the old values in the remaining fields).

MALLTUpdateByID

```
*****
/* Purpose: This example updates a pirep. */
/*          1. Connect to database. */
/*          2. Retreive a specific observation. */
/*          3. Update the observation. */
/*          4. Disconnect from the database. */
*/
/* Note: Two compiler Errors will occur. */
/*       obRef.nObjectID = ; Set to object ID ouput with ingest example */
/*       obRef.szTableName; Set to Table Name ouput with ingest example */
/* Tips: The update API resets the obRef field with a new objectID and */
/*       possibly a new table. */
*/
/* Responsible Organization: */
/*       Integrated Performance Decisions, Monterey Sector (RLJones) */
*/
*****
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "MALLTAPI.h"
main()
{
    MALLTOREFERENCE    obRef;
    MALLTRET           malltRet;
    MALLTObservation   theOb;
    /*
     * Connect to the default "mdl1t_db" database */
    malltRet = MALLTConnect();
    if(!CheckError(malltRet) )
    {
        /* Update the remark turbulence fields */
        obRef.nObjectID = 1;
        strcpy(obRef.szTableName,"pr1998062922");
        malltRet = MALLTGetByID(obRef,&theOb);
        if(!CheckError(malltRet) )
        {
            strcpy(theOb.ob.pirep.szRemarks,"Sure is smooth up here!");
            theOb.ob.pirep.sTurbulence=0;
            malltRet = MALLTUpdateByID(obRef,&theOb);
        }
    }
    printf("New Object ID %d \n", theOb.obRef.nObjectID);
```

```
    printf("New Table Name is %s \n",theOb.obRef.szTableName);
    MALLTDisconnect();
}
```

4.1.5 Deleting LLT Observations From the Database

Two methods are provided for deleting LLT observations from the database:

- MALLTDeleteByQuery
- MALLTDeleteByID

The MALLTDeleteByQuery method takes as input a MALLTCATALOGQUERY structure containing criteria for observations to be deleted. It deletes all observations that matched the input criteria. The function return is a MALLTRET status structure.

The second method deleting LLT observations is MALLTDeleteByID. This method takes as input a MALLTOBREFERENCE structure identifying the observation to be deleted. This method requires knowledge of the reference data for the record to be deleted, which is typically found by calling MALLTCatalog to retrieve a list of LLT observations meeting specific criteria. Note that if MALLTCatalog is called, the linked list it returns must be freed afterward.

```
*****
/* Purpose: This example deletes a single ob by ID. */
/*          1. Connect to database. */
/*          2. Specify an observation to delete */
/*          3. Delete the observation. */
/*          4. Disconnect from the database. */
/*
/* Note: You will need to fill obRef.nObjectID and obRef.szTableName to */
/*       an object id and table name that was ingested via the test */
/*       You can determine on by using the catalog testdriver or example. */
/* Responsible Organization: */
/*          Integrated Performance Decisions, Monterey Sector (RLJones) */
/*
*****
```

```
#include "MALLTAPI.h"
main()
{
    MALLTOBREFERENCE obRef;
    MALLTOBSERVATION anOb;
    MALLTRET malltRet;
    malltRet = MALLTConnect();
    if (!malltRet.nStatus)
    {
        obRef.nObjectID = 5;
        strcpy(obRef.szTableName,"tt1998062922");
        malltRet = MALLTDeleteByID( obRef );
    }
    if (!malltRet.nStatus)
    {
        printf("MALLTRET: status %d ",malltRet.nStatus);
        printf("SQLSTATE %s \n",malltRet.szSQLState);
        printf("Error Message: %s \n",malltRet.szErrorMessage);
```

```
    }
    malltRet = MALLTDisconnect();
}
```

MALLTDeleteByQuery

```
*****
/* Purpose: This example deletes obs by a specified criteria.          */
/*           1. Connect to database.                                     */
/*           2. Specify a delete criteria                            */
/*           3. Delete the observations.                           */
/*           4. Disconnect from the database.                         */
/*           */                                                 */
/* Responsible Organization:                                         */
/*     Integrated Performance Decisions, Monterey Sector (RLJones)   */
/*     */                                                 */
*****
```

```
#include <stdio.h>
#include <stdlib.h>
#include "MALLTAPI.h"
main()
{
    MALLTOBCATALOGQUERY dq;
    MALLTRET mr;
/* define desired observation type and subtype                      */
    dq.sObType = MALLT_UPPERAIR_TEMP; /* obtype: UPPER AIR TEMP      */
    dq.sObSubType = MALLT_FIXED_LAND; /* subtype: FIXED LAND REPORT  */

    /* Wildcard WMO ID */
    dq.nSiteID = MALLTsetInt2Null();
    /* wildcard the call sign */
    dq.szSiteName[0] = 0;
    dq.nDataCatagory = MALLTsetInt2Null();
    dq.nQualityIndicator = MALLTsetInt2Null();

    /* Query within given area*/
    dq.geoArea.rsNorthLat = 80.0;
    dq.geoArea.rsSouthLat = 20.0;
    dq.geoArea.rsWestLon = 0.0;
    dq.geoArea.rsEastLon = 80.0;

    /* Set time range to purge out obs that are more than 24 hours hold*/
    dq.nMinTime = 0;
    dq.nMaxTime = time(NULL)-24*3600;
    /* Ignore Time that obs were ingested */
    dq.nMinIngestTime = MALLTsetInt2NULL();
    dq.nMaxIngestTime = MALLTsetInt2NULL();

/* Connect to the database */
    mr = MALLTConnect();

    if (!mr.nStatus)
    {

        mr=MALLTDeleteByQuery(dq);
    }
    if(mr.nStatus)
    {
        printf("MALLTRET: status %d ",mr.nStatus);
        printf("SQLSTATE %s \n",mr.szSQLState);
        printf("Error Message: %s \n",mr.szErrorMessage);
    }
}
```

```
    printf("SQLSTATE %s \n",mr.szSQLState);
    printf("Error Message: %s \n",mr.szErrorMessage);
}
mr = MALLTDisconnect();
}
```

4.1.6 Wildcards and NULL Values

When retrieving data from the database, it will generally be necessary to wildcard certain fields that are not useful in a particular query. For example, if a query is being performed within a geographic location, the Site IDs and Site Name are not necessary. Another case where fields may need to be set to NULL is when ingesting observation and some fields are not reported. Use the following functions to set fields to NULL.

```
double MALLTsetDouble2Null (void);
float MALLTsetFloat2Null (void);
short MALLTsetShort2Null (void);
int MALLTsetInt2Null (void);
long MALLTsetLong2Null (void);
```

These functions are detailed in the MALLT APIRM, Section 7.2.

Example:

The following example illustrates a catalog query to retrieve all ship reports within a specified area for the last hour.

```
#include<limits.h>
#include<stdlib.h>
#include<time.h>
#include<MALLTAPI.h>

main()
{
    MALLTOBCATALOGQUERY cq;
    MALLTLINKEDLIST *pll,catalog;
    PMALLTOBCATALOG pOC;
    PMALLTFCSTOBECATALOG pFOC;
    MALLTRET mr;
    int nNumObs;

    MALLTConnect();
    cq.nSiteID =
    cq.nDataCatagory =
    cq.nQualityIndicator = MALLTsetInt2Null ();
    /* query for current time minus subtract */
    /* 1 hour */
    cq.nMinTime = time(NULL)-3600;
    cq.nMaxTime = time(NULL); /* Current Time */
    cq.sObType = MALLTsetShort2Null();
    cq.sObSubType = MALLT_SHIP;
    /*Wild card for character strings is a zero */
    /* length null terminated string */
```

```
cq.szSiteName[0]=0;
cq.geoArea.rsNorthLat = MALLTsetFloat2Null();
cq.geoArea.rsSouthLat = MALLTsetFloat2Null();
cq.geoArea.rsWestLon = MALLTsetFloat2Null();
cq.geoArea.rsEastLon = MALLTsetFloat2Null();
mr = MALLTCatalog( cq,&nNumObs, &catalog);

if(!mr.nStatus)
{
    for(pll=&catalog; pll && pll->data; pll = pll->next)
    {
        pOC = pll->data;
        printf("\nObject ID      : %d\n",pOC->obRef.nObjectID);
        printf("Table          : %s\n",pOC->obRef.szTableName);
        printf("Ob Type id    : %hd\n",pOC->obType.sObType);
        printf("Ob Type       : %s\n",pOC->szObTypeName);
        printf("Ob Sub Type id: %hd\n",pOC->obType.sObSubType);
        printf("Ob Sub Type   : %s\n",pOC->szObSubTypeName);
        printf("SiteName       : %s\n",pOC->szSiteName);
        printf("ReceiptMethod  : %s\n",pOC->szReceiptMethod);
        if(pOC->obType.sObType == MALLT_AERODROME_OB &&
           pOC->obType.sObSubType == MALLT_TAF_REPORT )
        {
            pFOC = (PMALLTFCSTOBATALOG)pll->data;
            printf("Flag              :%d\n",pFOC->nDataCatagory);
            printf("nQualityIndicator :%d\n",pFOC->nQualityIndicator);
            printf("Latitude         :%f\n",pFOC->geoLoc.rsLatitude);
            printf("Longitude        :%f\n",pFOC->geoLoc.rsLongitude);
            printf("basetime(n)      :%d\n",pFOC->baseTime.lTime);
            printf("basetime(c)      :%s\n",pFOC->baseTime.szTime);
            printf("beginvalid(n)    :%d\n",pFOC->beginvalidtime.lTime);
            printf("beginvalid(c)    :%s\n",pFOC->beginvalidtime.szTime);
        }
        else
        {
            printf("Name ID          :%d \n",pOC->nSiteID);
            printf("Flag              :%d \n",pOC->nDataCatagory);
            printf("nQualityIndicator :%d \n",pOC->nQualityIndicator);
            printf("Latitude         :%f \n",pOC->geoLoc.rsLatitude);
            printf("Longitude        :%f \n",pOC->geoLoc.rsLongitude);
            printf("Obtime(n)        :%d \n",pOC->baseTime.lTime);
            printf("Obtime(c)        :%s \n",pOC->baseTime.szTime);
        }
    }
}
else
{
    printf("MALLTRET: status %d ",mr.nStatus);
    printf("SQLSTATE %s \n",mr.szSQLState);
    printf("Error Message: %s \n",mr.szErrorMessage);
}

MALLTDisconnect();
exit(0);
}
```

4.1.7 Checking for NULL Values

To determine if a field is NULL (missing), use the following functions:

```
int MALLTisDoubleNull (double rd);
int MALLTisFloatNull (float rs);
int MALLTisShortNull (short s);
int MALLTisIntNull (int n);
int MALLTisLongNull (long l);
```

A value of 1 is returned if the field is NULL and 0 if it is not.

4.2 Building Applications Using the MALLT Libraries

This section contains four makefile examples that can be used to build the software using the MALLT APIs. Archive (.a) and shared libraries (.sl) makefiles are provided for the HP-UX environment. Examples using libraries (.lib) and dynamic link libraries (.dll) are provided for the Windows NT environment.

4.2.1 Makefile Using the Dynamic MALLT Libraries on HP-UX

```
CFILE    = \
          catalog.c \
          delByID.c \
          delByQuery.c \
          getByid.c \
          getByquery.c \
          ingest.c \
          update.c

OFILE    = $(CFILE:.c=.o)
PROGRAMS= $(OFILE:.o=)

LIBS     = \
          $(MALLT_HOME)/bin/libMALLTAPI.sl \
          $(MALLT_HOME)/bin/libMALLTDsmgr.sl \
          $(MALLT_HOME)/bin/libMALLTKernel.sl \
          $(MALLT_HOME)/bin/libMALLTUtils.sl

INCLUDES = -I$(MALLT_HOME)/include

# other definitions (must run under sh shell)

SHELL   = /bin/sh
RM      = /bin/rm -f

LD      = esql
LDFLAGS = $(LIBS) +z common.o
CC      = cc

all : common.o $(OFILE) $(PROGRAMS)
```

```
clean :  
        $(RM) $(OfFile) $(PROGRAMS) commons.o core  
  
common.o:  
        $(CC) -c $(CFLAGS) common.c -o common.o  
  
.c.o :  
        $(CC) -c $(CFLAGS) $*.c -o $*.o  
  
.o:  
        $(LD) $*.o $(LDFLAGS) -o sl$*
```

4.2.2 Makefile Using the Static MALLT Libraries on HP-UX

```
CFILE = \  
catalog.c\  
delByID.c\  
delByQuery.c\  
getbyid.c\  
getbyquery.c\  
ingest.c\  
update.c  
  
OfFile = $(CFILE:.c=.o)  
PROGRAMS= $(OfFile:.o=)  
  
LIBS = \  
$(MALLT_HOME)/lib/libMALLTAPI.a\  
$(MALLT_HOME)/lib/libMALLTDsmgr.a\  
$(MALLT_HOME)/lib/libMALLTKernel.a\  
$(MALLT_HOME)/lib/libMALLTUtils.a  
  
INCLUDES = -I$(MALLT_HOME)/include  
  
# other definitions (must run under sh shell)  
  
SHELL = /bin/sh  
RM = /bin/rm -f  
  
LD = esql  
LDFLAGS = $(LIBS) common.o  
CC = cc  
  
all : common.o $(OfFile) $(PROGRAMS)  
  
clean :  
        $(RM) $(OfFile) $(PROGRAMS) commons.o core  
  
common.o:  
        $(CC) -c $(CFLAGS) common.c -o common.o  
  
.c.o :  
        $(CC) -c $(CFLAGS) $*.c -o $*.o  
  
.o:  
        $(LD) $*.o $(LDFLAGS) -o sl$*
```

4.2.3 Makefile Using the Dynamic Link MALLT Libraries on Windows NT

```
MAKEFILE = make.dll

MKDIR = mkdir
CP = copy

PROGRAM= catalog.exe\
        delByID.exe\
        delByQuery.exe\
        getbyid.exe\
        getbyquery.exe\
        ingest.exe\
        update.exe

CFILES = catalog.c\
        delByID.c\
        delByQuery.c\
        getbyid.c\
        getbyquery.c\
        ingest.c\
        update.c

OFILES = $(CFILES:.c=.obj)

LIBS= \
    $(MALLT_HOME)\bin\malltapi.lib \
    $(MALLT_HOME)\bin\malltdsmgr.lib \
    $(MALLT_HOME)\bin\malltkernel.lib \
    $(MALLT_HOME)\bin\malltutils.lib

INCLUDES = -I$(MALLT_HOME)\include

LDFLAGS = $(LIBS) common.obj
# other definitions

RM = -@del /Q /F /S

LD = esql
CFLAGS = -c $(INCLUDES) -D_MDBLL -D_WIN32 -DWIN32 -W3 -DSTRICT -D__STDC__=0

#####
# TARGETS #####
#####

all: common.obj $(OFILE) $(PROGRAM)

clean :
    $(RM) $(OFILES) *.lnk *.map

.c.exe :
    $(CC) $(CFLAGS) $*.c -o $*.obj
    $(LD) $*.obj $(LDFLAGS) -o dll$*

common.obj:
    $(CC) $(CFLAGS) $*.c -o $*.obj
~
```

4.2.4 Makefile Using the Static MALLT Libraries on Windows NT

```
MAKEFILE = make.lib

MKDIR = mkdir
CP = copy

PROGRAM= catalog.exe\
        delByID.exe\
        delByQuery.exe\
        getbyid.exe\
        getbyquery.exe\
        ingest.exe\
        update.exe

CFILES = catalog.c\
        delByID.c\
        delByQuery.c\
        getbyid.c\
        getbyquery.c\
        ingest.c\
        update.c

OFILES = $(CFILES:.c=.obj)

LIBS= \
      $(MALLT_HOME)\lib\malltapi.lib \
LIBS= \
      $(MALLT_HOME)\lib\malltapi.lib \
      $(MALLT_HOME)\lib\malltdsmgr.lib \
      $(MALLT_HOME)\lib\malltkernel.lib \
      $(MALLT_HOME)\lib\malltutils.lib

INCLUDES = -I$(MALLT_HOME)\include

LDFLAGS = $(LIBS) common.obj
# other definitions

RM = -@del /Q /F /S

LD = esql
CFLAGS = -c $(INCLUDES) -D_WIN32 -DWIN32 -W3 -DSTRICT -D__STDC__=0

#####
##### TARGETS #####
#####

all: common.obj $(OFILES) $(PROGRAM)

clean :
    $(RM) $(OFILES) *.lnk *.map

.c.exe :
    $(CC) $(CFLAGS) $*.c -o $*.obj
    $(LD) $*.obj $(LDFLAGS) -o lib$*

common.obj:
    $(CC) $(CFLAGS) $*.c -o $*.obj
```

(This page intentionally left blank.)

5 CUSTOMIZING SEGMENTS

This section is tailored out.

(This page intentionally left blank.)

6 NOTES

6.1 Glossary of Acronyms

AESS	Allied Environmental Support System
ANSI	American National Standards Institute
API	Application Program Interface
APIRM	API Reference Manual
COE	Common Operating Environment
DII	Defense Information Infrastructure
GCCS	Global Command and Control System
IC4ISR	Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance
IMOSS	Interim Mobile Oceanographic Support System
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MALLT	LLT Observation API Segment
MDLLT	LLT Observation Database Segment
METOC	Meteorology and Oceanography
MIDDS	Meteorological Integrated Data Display System
NITES	Navy Integrated Tactical Environmental Subsystem

PC	Personal Computer
PM	Programming Manual
PS	Performance Specification
RDBMS	Relational Database Management System
SQL	Structured Query Language
TAF	Terminal Aerodrome Forecast
TEDS	Tactical Environmental Data System
TESS(NC)	Tactical Environmental Support System Next Century

Appendix A - Valid Values, WMO Symbolic Code, and Units for MALLT Structures

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE

ipd4400malltpmTES-10

(This page intentionally left blank.)

Structure	Field	WMO 306 Symbolic Code	Unit	Field Validation Rule
MALLTAERODROMELOUDS	sCloudHeight	h _s h _s	meters/30	between (0 and 999)
MALLTAERODROMELOUDS	szCloudAmount	N _s N _s N _s	N/A	IN ('SKC,' 'skc,' 'FEW,' 'few,' 'SCT,' 'sct,' 'BKN,' 'bkn,' 'OVC,' 'ovc')
MALLTAERODROMELOUDS	szCloudType	N/A (appended to end of cloud group)	N/A	IN ('CU,' 'cu,' 'CB,' 'cb,' '')
MALLTAERODROMEQP	szDescriptor	w'w'	N/A (Code table 4678)	IN (',' 'MI,' 'BL,' 'BC,' 'SH,' 'PR,' 'DR,' 'TS,' 'FZ')
MALLTAERODROMEQP	szIntensity	w'w'	N/A (Code table 4678)	IN ('-,' '+,' ',' 'VC,' 'vc')
MALLTAERODROMEQP	szPhenomena	w'w'	N/A (Code table 4678)	In (',' 'DZ,' 'IC,' 'RA,' 'PE,' 'SN,' 'GR,' 'SG,' 'GS,' 'UP,' 'BR,' 'SA,' 'FG,' 'HZ,' 'FU,' 'PY,' 'VA,' 'DU,' 'SQ,' 'SS,' 'DS,' 'PO,' 'FC')
MALLTAEROFORECAST	sProbability	C ₂ C ₂	per cent	between (0 and 100)
MALLTAEROFORECAST	szFcstIndicator	TT	N/A	IN ('FM,' 'TO,' 'TL')
MALLTAEROFORECAST	szTrend	TTTTT	N/A	IN ('PROB,' 'TEMPO,' 'BECMG')
MALLTBATHYSOUNDING	rsCurrentSpeed	V _c V _c	knots	between (0 and 30)
MALLTBATHYSOUNDING	rsDepth	Z _n Z _n (BATHY) or Z _n Z _n Z _n Z _n (TESAC)	meters	between (-100 and 12000)
MALLTBATHYSOUNDING	rsSalinity	S _n S _n S _n S _n	parts per thousand	between (0 and 45)
MALLTBATHYSOUNDING	rsWaterTemp	T _n T _n T _n (BATHY) or T _n T _n T _n T _n (TESAC)	Degrees Celsius	between (-2. and 40.)
MALLTBATHYSOUNDING	sCurrentDir	D _c D _c	Degrees True	between (0 and 359)
MALLTBUOY	rsMSLPressure	PPPP	hectopascals	between (830 and 1090)
MALLTBUOY	rsPressureTendAmt	ppp	hectopascals	between (0 and 99)
MALLTBUOY	rsWavePeriodByInst	P _{wa} P _{wa}	seconds	between (0 and 99.9)
MALLTBUOY	sDriftSpeed	V _B V _B	cm/sec	between (0 AND 9999)
MALLTBUOYSOUNDING	rsDepth	Z _d Z _d Z _d	meters	between (0 and 12000)
MALLTBUOYSOUNDING	rsSalimit	S _n S _n S _n S _n	parts per thousand	between (0 and 45)
MALLTBUOYSOUNDING	rsWaterTemp	T _n T _n T _n T _n	degrees Celsius	between (-2 and 40)
MALLTCLOUDDATA	rsCloudHeight	h _s h _s	N/A (Code Table 1677)	height>0
MALLTCLOUDDATA	rsHeightAboveSfc	h	N/A (Code Table 1600)	heightabovesfc>0
MALLTCLOUDDATA	szHighCloud	C _H	N/A (Code Table 0509)	IN ('0,' '1,' '2,' '3,' '4,' '5,' '6,' '7,'

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE
ipd4400malltpmTES-10

Structure	Field	WMO 306 Symbolic Code	Unit	Field Validation Rule
MALLTCLOUDDATA	szLowCloud	C _L	N/A (Code Table 0513)	IN ('0,' '1,' '2,' '3,' '4,' '5,' '6,' '7,'
MALLTCLOUDDATA	szMidCloud	C _M	N/A (Code Table 0515)	IN ('0,' '1,' '2,' '3,' '4,' '5,' '6,' '7,'
MALLTCONVCOND	nHeight	N/A	meters	between (0 AND 150000)
MALLTCONVCOND	rsPressure	N/A	hectopascals	between (0 and 1100)
MALLTCONVCOND	rsTemperature	N/A	degrees Celsius	between (-110 and 63)
MALLTCONVCOND	szType	N/A	N/A	in ('a,' 'A,' 'm,' 'M')
MALLTEVAPORATIONHT	nEvapDuctHeight	N/A	meters	between (0 and 150000)
MALLTEVAPORATIONHT	rsDepointTemp	N/A	degrees Celsius	between (-110 and 63)
MALLTEVAPORATIONHT	rsSst	N/A	degrees Celsius	between (-5 and 45)
MALLTEVAPORATIONHT	rsSurfaceAirTemp	N/A	degrees Celsius	between (-110.surfaceairtemp<=63.)
MALLTEVAPORATIONHT	rsWindSpeed	N/A	meters/second	between (0 AND 300)
MALLTEVAPORATIONHT	sRelativeHumidy	N/A	per cent	between (0 AND 100)
MALLTFCSTOBCATALOG	baseTime	N/A	seconds since 1/1/1970	
MALLTFCSTOBCATALOG	begininvalidtime	N/A	seconds since 1/1/1970	
MALLTFCSTOBCATALOG	endvalidtime	N/A	seconds since 1/1/1970	
MALLTGENERALCONDITIONS	rsAirTemperature	TTT	degrees Celsius	
MALLTGENERALCONDITIONS	rsAltitude	h _d h _d h _d	Hundreds of feet	
MALLTGENERALCONDITIONS	rsPressure	P ₀ P ₀ P ₀ P ₀	hectopascals	between (-110. AND 63)
MALLTGENERALCONDITIONS	rsWindSpeed	ff	meters/second	
MALLTGENERALCONDITIONS	sWindDirection	dd	degrees True	between (450. and 1100.)
MALLTGENERICID	szSiteName	D.....D or CCCC	N/A	between (0. AND 300)
MALLTGENERICID	nSiteID	Iiiii	N/A	between (0 AND 359) OR (winddirection=-1)
MALLTFCSTOBCATALOG	geoLoc	N/A	N/A (structure)	between (0 AND 999999)
MALLTFCSTOBCATALOG	szSiteName[16]	N/A	N/A	
MALLTGEOPBOUNDS	rsEastLon	N/A	degrees	
MALLTGEOPBOUNDS	rsNorthLat	N/A	degrees	
MALLTGEOPBOUNDS	rsSouthLat	N/A	degrees	
MALLTGEOPBOUNDS	rsWestLon	N/A	degrees	
MALLTGEOPPOINT	rsLatitude	L _a L _a L _a	degrees	between (-90. AND 90.)
MALLTGEOPPOINT	rsLongitude	L _o L _o L _o L _o	degrees	between (-180. AND 180.)
MALLTMETARFIELDS	nWindGusts	f _m f _m	meters/second	between (0. AND 200.)
MALLTMETARFIELDS	rsAirTemperature	T'T'	degrees Celsius	between (-110. AND 63.)

MALLTMETARFIELDS	szWindVaribility	V	N/A	IN ('v,' 'V,' '')
MALLTMETOCTIME	lTime	YYGGgg or YYGG	epoch time	
MALLTMETOCTIME	szTime[32]	YYGGgg or YYGG	N/A	
MALLTOBAOIS	nAOIID	N/A	N/A	
MALLTOBAOIS	szAoiName[64]	N/A	N/A	
MALLTOBCATALOG	baseTime	YYGGgg or YYGG	epoch time	
MALLTOBCATALOG	geoLoc	L _a L _a L _a and L _o L _o L _o L _o	degrees	
MALLTOBCATALOG	nSiteID	Iiiii	N/A	
MALLTOBCATALOG	szSiteName[16]	D.....D or CCCC	N/A	
MALLTPIREP	szMoistureType	N/A	N/A	IN ('0,' '1,' '2')
MALLTPIREP	szNavSystem	N/A	N/A	IN ('0,' '1')
MALLTPIREP	szPhaseOfFlight	N/A	N/A	IN ('LVR,' 'LVW,' 'ASC,' 'DES,' '')
MALLTROCKETSOUNDING	nHeight	HH, h _n h _n h _n	meters	between (0 AND 150000)

MALLTUAMMWINDS	rsWind1KMAboveMax	$v_a v_a$	meters/second	between (0. AND 300.)
MALLTUAMMWINDS	rsWind1KMBelowMax	$v_b v_b$	meters/second	between (0. AND 300.)
MALLTUAMMWINDS	sIRCorrection	c_T	N/A (Code Table 0659)	between (0 AND 7)
MALLTUAMMWINDS	sTrackingSystem	a_4	N/A (Code Table 0265)	between (0 AND 99)

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE
ipd4400malltpmTES-10

Structure	Field	WMO 306 Symbolic Code	Unit	Field Validation Rule
MALLTUAPROFILE	nCrossIndex	N/A	N/A	between (0 AND 70)
MALLTUAPROFILE	nEquilLevel	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	nEquilTemp	N/A	degrees Celsius	between (-110 AND 63)
MALLTUAPROFILE	nHeightContrailBPO	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	nHeightContrailBPRB	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	nHeightContrailTPO	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	nHeightContrailTPRB	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	nKIndex	N/A	N/A	between (0 AND 50)
MALLTUAPROFILE	nLiftedIndex	N/A	N/A	between (-10 AND 10)
MALLTUAPROFILE	nShowalterIndex	N/A	N/A	between (-10 AND 10)
MALLTUAPROFILE	nSweatIndex	N/A	N/A	between (0 AND 500)
MALLTUAPROFILE	nTotalIndex	N/A	N/A	between (9 AND 110)
MALLTUAPROFILE	nVertIndex;	N/A	N/A	between (0 AND 40)
MALLTUAPROFILE	nWetBulbZHeight	N/A	meters	between (0 AND 150000)
MALLTUAPROFILE	rsConvTemp	N/A	degrees Celsius	between (-110. AND 63.)
MALLTUAPROFILE	rsMaxConWndGustSp	N/A	meters/second	between (0 AND 300.)
MALLTUAPROFILE	rsMaxHailSize	N/A	millimeters	between (0 AND 20.)
MALLTUAPROFILE	rsPrecipWatCon	N/A	millimeters	between (0 AND 80.)
MALLTUAPROFILE	sMaxConWindGustDir	N/A	degrees	between (0 AND 359)
MALLTUAPROFILESOUNDING	nHeight	N/A	meters	between (0 AND 150000)
MALLTUAPROFILESOUNDING	nMUnits	N/A	N/A	between (0 AND 7000)
MALLTUAPROFILESOUNDING	nPotentialTemp	N/A	degrees Celsius	between (-110. AND 63.)
MALLTUAPROFILESOUNDING	rsVerticalWindShear	N/A	meters/second/100 meters	between (0 AND 300.)
MALLTUAPROFILESOUNDING	sHeightType	N/A	N/A	between (-1 AND 6)
MALLTUAPROFILESOUNDING	sRichsNumber	N/A	N/A	between (-5 AND 5)
MALLTUAPROFILESOUNDING	sRLType	N/A	N/A	between (1 AND 5)
MALLTUAPROFILESOUNDING	sTurbulenceIntsty	B	N/A (Code Table 0300)	between (0 AND 9)
MALLTUAPROFILESOUNDING	sTurbulenceProb	N/A	per cent	between (0 AND 100)
MALLTUATEMP SOUNDING	sHeightType	N/A (determined by report section)	N/A	IN (-1, 1, 2, 3, 4, 5, 6)
MALLTUATTI	rsSst	T _s T _w T _w	degrees Celsius	between (-5. AND 45.)
MALLTUATTI	sHeightLowestIce	h _j h _i h _i	N/A (Code Table 1690)	between (400 AND 50000)
MALLTUATTI	sHeightLowestTurb	h _B h _B h _B	N/A (Code Table 1690)	IN (0, 2, 4, 6)
MALLTUAWSOUNDING	sHeightType	N/A (determined by report section)	N/A	IN (-1, 1, 2, 3, 4, 5, 6)

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE

ipd4400malltpmTES-10

(This page intentionally left blank.)